Wireless Networking Testbed and Emulator (WiNeTestEr)*

Kiruba S. Subramani,
Joseph D. Beshay,
Niranjan Mahabaleshwar,
Ehsan Nourbakhsh,
Brooks McMillin,
Bhaskar Banerjee,
Ravi Prakash
The University of Texas at Dallas
Richardson, Texas, USA
{kiruba.subramani, joseph.beshay,
niranjan.mahabaleshwar, ehsaan,
brooks.mcmillin, bhaskar.banerjee,
ravip} @utdallas.edu

Yongjiu Du, Pengda Huang,
Tianzuo Xi, Yang You,
Joseph D. Camp,
Ping Gui, Dinesh Rajan
Southern Methodist University
Dallas, Texas, USA
{ydu, phuang, txi,yyou, camp,
pgui, rajand} @smu.edu

Jinghong Chen
The University of Arizona
Tucson, Arizona, USA
{jhchen}@email.arizona.edu

ABSTRACT

Repeatability, isolation and accuracy are the most desired factors while testing wireless devices. However, they cannot be guaranteed by traditional drive tests. Channel emulators play a major role in filling these gaps in testing. In this paper we present an efficient channel emulator which is better than existing commercial products in terms of cost, remote access, support for complex network topologies and scalability. We present the hardware and software architecture of our channel emulator and describe the experiments we conducted to evaluate its performance against a commercial channel emulator.

Keywords

Wireless; RF; Channel Emulation; Testbed

1. INTRODUCTION

The ability to conduct repeatable experiments is crucial to the development of wireless devices and protocols. Most of the time, researchers make simplifying assumptions about the nature of their test environment and the experiment control procedures. However these assumptions do not always hold good [1] making it harder to isolate device/protocol performance from environmental effects.

The current wireless networking testbeds use a wide range of approaches, varying from fully software-simulated testbeds like ns-3 [2] to real hardware running in Faraday cages. However, the two extremes have their respective limitations. Simulators are easy to implement but they are limited by the models provided in software. Different simulators might yield different results due to the assumptions and simulation techniques used [3]. On the other hand using hardware in Faraday cages is not always affordable and it does not allow flexibility to test complicated scenarios involving mobility or signal reflections. The balance between the two extremes is provided by channel emulators.

The simplest form of channel emulation is achieved by using shielded RF cables between the transceivers and a programmable attenuator in the signal chain. An example of an emulator based on this design was ASSERT [4], developed by our group. By increasing (or decreasing) the attenuation we simulated the transceivers moving apart (or moving closer). The rate at which attenuation was varied corresponded to the relative speed of the transceivers. This scenario covered the fading effect of wireless channels.

However, wireless transmissions are not only affected by fading but also by multiple reflections of the same signal from obstacles in the surrounding environment (multi-path effects). Devices with multiple antennas (MIMO) exploit these signal reflections to achieve better throughput. Accurate emulation of multi-path effects requires creating multiple copies of the transmitted signal with different time delays (phases). This cannot be achieved by attenuators. Instead it is done by digitizing the signal and manipulating it using digital signal processing (DSP). The resulting digital signal is converted back to analog.

Commercial solutions exist for emulating environments with multi-path effects [5] [6]. However they are prohibitively expensive and are limited to simulating environments with 2 pairs of devices or less. Commercial channel emulators are thus impractical for researchers who are usually cost-constrained and interested in experiments that involve the interaction (and interference) between multiple devices with a higher degree of connectivity. As a result, researchers sought to develop their own channel emulators that could achieve multipath effects for multiple devices while maintaining relatively low cost. An example is the work in [7] which uses a single field programmable gate array (FPGA)

*This work is supported in part by the National Science Foundation MRI program under grant numbers 1040422 and 1040429.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
MSWiM '14, September 21 - 26 2014, Montreal, QC, Canada
Copyright 2014 ACM 978-1-4503-3030-5/14/09 ...$15.00.
http://dx.doi.org/10.1145/2641798.2641809.

to simulate a 90Mhz-wide environment for up to 15 devices operating in the 2.4 GHz ISM band. The design in [7] cannot scale due to the FPGA resource constraints.

In this paper, we present our Wireless Networking Testbed and Emulator (WiNeTestEr) which is designed to simulate 100-Mhz-wide environments with multipath in the 2.4 GHz ISM band. The main features of WiNeTestEr are:

- scalability: the system uses a distributed channel emulation algorithm running on multiple FPGAs so it can potentially scale to hundreds of nodes,
- remote access: a device control protocol allows the system to run experiments without onsite-operator intervention,
- concurrent experiments: the modular design allows for multiple independent experiments to be run by different users at the same time,
- multi technology support: experiments can be performed on different technology devices operating in their native frequencies (Bluetooth, WiFi, Zigbee, etc.). The design is flexible to allow adding more frequency bands in the future with minimal changes.
- full duplex channels: the channel between two devices is full duplex with support for non-reciprocal channel conditions i.e. the signal can experience a certain environment in one direction and a different one in the other.

WiNeTestEr's design is loosely based upon ASSERT [4]. ASSERT performs channel emulation in the 900 MHz ISM band using attenuation. Attenuators are used to control the transmitted signal strength to emulate the required channel conditions (deep fade, slow fade) but it cannot emulate multi-path effects. WiNeTestEr was developed to bridge this gap.

2. SYSTEM DESIGN

WiNeTestEr consists of two distinct networks; the control network and the RF network as shown in Figure 1. The control network is where the experiment control takes place. It consists of the Control PC and Wireless Open-Access Research Platform (WARP) v2.2 boards [8] connected by an Ethernet network. The RF network is where the channel emulation happens. It consists of a set of units under test (UUTs) interconnected across RF and combiner boards. All the connections in the RF network use shielded coaxial cables. The RF board has one input and three outputs, and is responsible for converting the input signal from analog to digital and passing it to the FPGA on the WARP board on which the RF board is mounted. The FPGA applies the desired channel conditions for each output. The RF board converts each signal back to analog to be output on the equivalent port. The combiner board is a passive board that is responsible for combining up to four signals into one to be sent to a UUT.

A unidirectional link from UUT A to UUT B is realized through seven steps as shown in Figure 1:

1. UUT A's output is connected to UUT A's RF board through a wideband duplexer (circulator).
2. RF board digitizes the signal and passes it to its WARP board's FPGA.

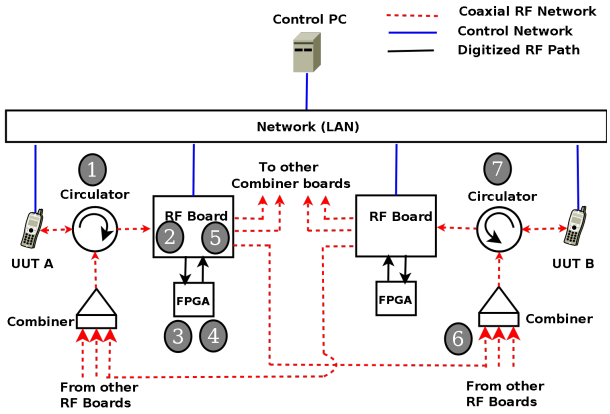

Figure 1: A simple topology of two UUTs connected over a single bidirectional link.

3. FPGA makes copies of the signal for each of the three outputs.
4. FPGA processes the digital signal based on the channel conditions set by the control PC for each output.
5. RF board converts the signals back to analog and outputs each of them on its port.
6. The equivalent RF board output is connected to UUT B's combiner board through a coaxial cable.
7. Combiner board passes the combined signal of all of its inputs to UUT B through the circulator.

Using the unidirectional link as a building block, it is possible to build any higher degree topology involving any number of UUTs. For example, a simple bidirectional link can be formed from two unidirectional links as shown in Figure 1. More details on topology formation are discussed in the implementation section.

An experiment is run by specifying the channel condition that should be applied to each link in the topology for a certain period of time. The Control PC orchestrates the experiment by applying the channel conditions to each RF board output and signaling the UUTs to start executing. Once the experiment expires, the Control PC collects the results from UUTs and resets the channel conditions.

In WiNeTestEr, the hardware and software modules go hand in hand in carrying out the experiments scheduled by the user. We introduce the hardware and software architecture of our system in sections 2.1 and 2.2 respectively.

2.1 Hardware architecture

Our solution is based on development boards by WARP Project [8]. The WARP development board version 2.2 has a Xilinx Virtex 4 FPGA, 2 PowerPC cores, external memory slots, a RAM slot and other peripheral connectivity solutions to interface with external boards and controlling devices. Each WARP board can house two RF boards which can be connected to two different input UUTs. Figure 2 shows the overall architecture of a WARP board with two RF boards mounted.

Channel emulation can be accurately performed in the digital domain. The signal transmitted by the UUT is down-converted to baseband and digitized. Once the digitized

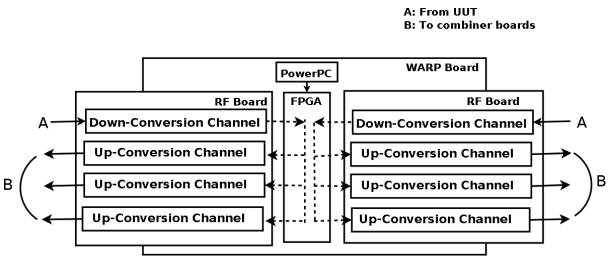

Figure 2: Architecture of a WARP board with two attached RF boards.

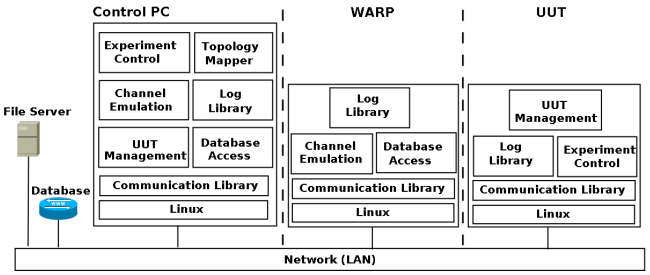

Figure 3: Software architecture.

signal is modified to reflect the desired channel emulation, it is converted back to analog and then up-converted. Channel emulation is done on the FPGA as instructed by a PowerPC processor.

The RF board consists of two signal chains, namely down-conversion and up-conversion. Each component on the board can be broadly classified as being part of one of the two chains. The down-conversion chain consists of digital step attenuator (DSA), quadrature demodulator, variable gain amplifier (VGA) and Analog to Digital Converter (ADC). The up-conversion chain consists of Digital to Analog Converter (DAC), VGA and quadrature modulator. Clock circuit present on each board helps in synchronizing data conversion operations. The clock circuit also provides reference input to Phase Locked Loop (PLL) which is used in generating Local Oscillator (LO) signal for down-conversion and up-conversion operations. As shown in Figure 2, the FPGA creates three identical copies of the original signal transmitted by the UUT. These signals can be independently modified as instructed by the PowerPC processor to emulate the desired environment. Each signal corresponds to an output port that will eventually be connected to a UUT via the combiner board.

2.2 Software architecture

The heart of WiNeTestEr's software architecture is the Control PC. It serves as the proxy between the user and other system components. The Control PC maintains a database of the experiment information as well as the hardware resources available in the system (WARP boards, RF boards, UUTs and cable connections). Users run a GUI application on their local machines and connect to the Control PC to retrieve the current system status, design their experiments and submit them for execution. To set the environment conditions, the Control PC sends the link parameters to the Power PC processors of the WARP boards to which the experiment's RF boards are connected.

The WARP board forwards the link parameters to the FPGA which applies them to the digitized signal received from the RF board. The FPGA could have directly read the link parameters from the Ethernet interface of the WARP board. However based on prior experiences from ASSERT, we decided to run embedded Linux on the WARP board (PowerPC processor) to provide a more advanced interface that can do more than receiving and parsing link parameters. Running Linux also allowed us to use high level libraries for log collection and experiment monitoring.

We designed a *UUT Management* protocol to be able to orchestrate experiments without human intervention. The UUT management protocol allows the Control PC to transfer execution images to UUTs to be used for experiments. The protocol also provides the interface to start, monitor, stop and collect experiment results from the UUTs. For the default setup, UUTs are Beagle Bone Black boards [9] which have ARM processors running Linux. Each board has a WiFi dongle (IEEE 802.11 b/g). Execution images are compressed archives containing ARM binaries and configuration files provided by the user. The UUT management protocol is used to transfer the specific archive to each UUT and control the experiment execution. This protocol however is an optional component of our system. Users who want to test their devices without exposing their binaries/configuration files may choose to use their own management protocol and only use WiNeTestEr for simulating the environment. They also have the option to implement the experiment control subset of the protocol without the image transfer. Our design is flexible to support multiple heterogeneous UUTs to exist and be one of multiple experiments running concurrently.

The software components running on the Control PC, WARP board, UUTs are shown in Figure 3. The following section outlines the purpose of each component and the main design decisions involved.

3. IMPLEMENTATION

3.1 RF Board

The RF board is responsible for frequency translation of the signal which allows channel emulation to be carried out on the FPGA based on environment conditions provided by the PowerPC processor. The architecture of the RF board is shown in Figure 4. RF interface to the board is provided by means of SMA jacks and coaxial cables. The use of coaxial cables minimizes interference among signals thereby increasing the robustness and repeatability of experiments. The RF board supports input signal power in the range of -25 dBm to +26 dBm and an output power of -20 dBm to -110 dBm. To achieve this target specification, a maximum attenuation of 136 dB is required from the design. This value is divided among multiple components on the board such as DSA, VGA and FPGA.

DSA performs the first stage of attenuation on the input signal. Owing to high signal strength of the input, there is a very high chance of it saturating the quadrature demodulator. Placing a DSA at the start of the chain helps restrain signal strength to reasonable values. Modern wireless standards use complex modulation schemes such as QPSK or QAM to achieve higher speed and lower error rate. To

support these standards, broadband quadrature modulator and demodulator with good performance specifications have been used in the up-conversion and down-conversion circuitry. For best-case performance of quadrature modulator, signal properties of In-phase (I) and Quadrature (Q) components need to be perfectly matched. In other words, a mismatch between these two components in terms of DC offset, gain and phase results in LO leakage and sideband issues thereby degrading the quality of output signal. Baseband VGAs are used to offset amplitude mismatches of I/Q signals. To obtain very low value of sideband signal, both gain and phase need to be carefully adjusted in the FPGA. Also, in down-conversion chain, VGA helps to amplify or attenuate the baseband signal to meet the dynamic range requirements of ADC. In WiNeTestEr, since the baseband signal has a bandwidth of 100 MHz, a sampling clock of 200 MHz is needed for the data converters to avoid aliasing. The clock circuit consists of a voltage controlled crystal oscillator (VCXO) as the clock source with clock distribution/divider IC's providing identical clocks to data converter chips (ADC and DAC).

Resolution of data converter ICs has significant impact on the performance of the system. During down-conversion, the demodulator outputs I/Q signals, requiring separate ADC to digitize each signal. Two 12-bit ADCs were used in the design to satisfy the signal to noise ratio (SNR) requirement of the receiver. Up-conversion, on the other hand, handles a wide range of output signal strength. Also, each RF board houses 3 up-conversion chains, with each chain having its own I/Q signals. Hence a dual 16-bit DAC was used on each chain to convert the signal back to analog domain.

Figure 5 shows the RF front end that was designed for WiNeTestEr. A single ended signal from the circulator is connected to the first SMA jack marking the input to the board. The other three SMA jacks correspond to the output from each up-conversion chain. The RF board supports three different power sources namely external adapter, WARP FPGA board and screw terminal. Selected source is regulated using low-dropout regulators (LDO) to generate 5V, 3.3V and 1.8V analog and digital supplies. Utmost care has been taken to reduce supply noise with the help of global

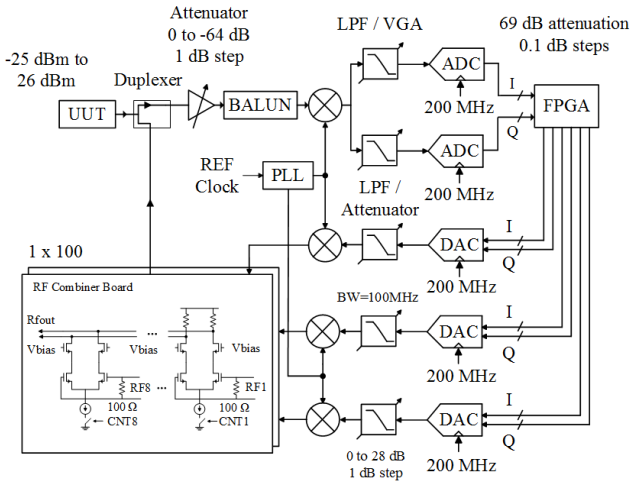

Figure 4: RF board architecture.

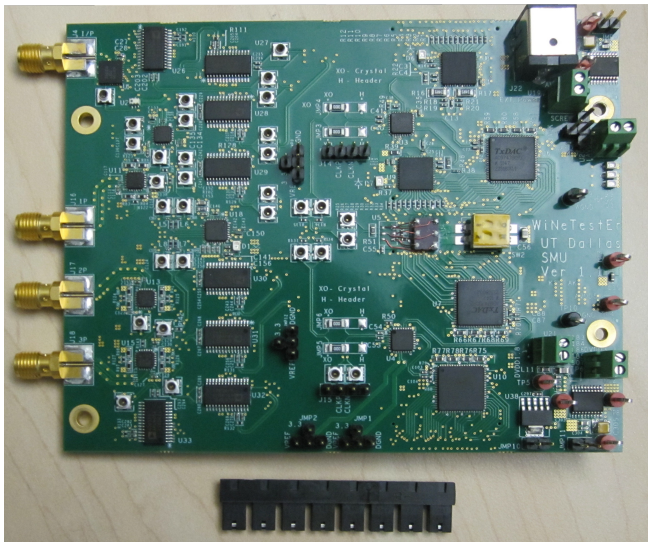

Figure 5: RF Board for WiNeTestEr.

and local decoupling capacitors and ferrite beads. Careful floor planning, layer management and termination techniques have been followed to obtain the best performance from the board.

3.2 WARP Board

We selected WARP v2.2 boards to house our RF boards due to the large number of pins available on 4 daughtercard slots. Each daughtercard slot has 124 pins which are routed to the dedicated I/O on the FPGA. With 4 such slots, we have 496 pins which are sufficient for communication between the FPGA and two RF boards. Xilinx Virtex 4 FPGA in WARP also has two PowerPC cores operating at 300 Mhz, one of which we used to run Linux. The Control PC communicates with the Linux via the Ethernet port on the WARP board.

Another notable feature in the WARP board is the System ACE chip. This chip connected to the CompactFlash slot on the board can program the FPGA and load Linux to the PowerPC processor using a single image file which has both the FPGA image and the Linux kernel. The System ACE chip reads these files from a FAT16 formatted CompactFlash card.

3.3 Combiner Board

The Combiner board is responsible for combining RF signals from different boards to implement multi-user interference. As shown in Figure 6, the combiner board contains four commercial low-noise amplifiers (LNA), four RF switches, and a custom designed 4-to-1 active combiner chip. The combiner chip uses active method to realize wideband RF signal combining and has a small chip area (1 mm × 1 mm, pads limited). The LNA is used to improve noise figure and sensitivity of the combiner board. When the input signal level is high, RF switches are used to bypass the LNA to improve the linearity of the combiner board.

3.4 Channel Emulation

The baseband channel emulation has been implemented on the FPGA and its architecture is shown in Figure 7. Each

tap has a delay unit(r_x) controlled by a user-defined value, followed by scaling based on fading parameters ($\rho_x(t)\phi_x(t)$) where x ∈ (0,6). The resulting signals from each tap are added together to emulate a multi-tap fading channel.

In WiNeTestEr, summation of sinusoids (SOS) method [10], [11] has been used to generate fading channel. This method of channel generation for emulation has been widely investigated in the past [12], [13], [14]. However, existing fading channel emulators demand large memory source to generate channels, which degrades the scalability. In this project, a novel structure to implement SOS based channel generation has been proposed [15]. With this structure, the generation of one Rayleigh channel consumes only 1 unit memory source (∼ RAMB16) of the available 376 in the FPGA. Besides reducing memory requirements, this work has also optimized word length selection and channel data update rate. Intuitively, larger bit width generates higher channel accuracy at the cost of hardware resources. Similar tradeoff exists between channel data generation rate and accuracy over time domain. In this project, optimization on the two terms (bit width and update rate) are performed [15], aiming at minimizing hardware resources at a certain channel accuracy level. With the reduction of memory consumption and optimization on the two terms, scalability is drastically improved on WiNeTestEr.

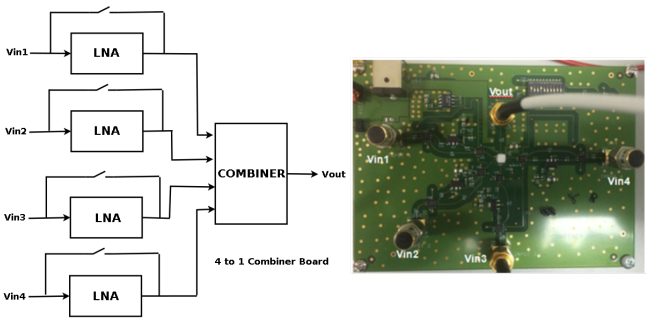

Figure 6: (Left) Architecture of the combiner board. (Right) Fabricated combiner board.

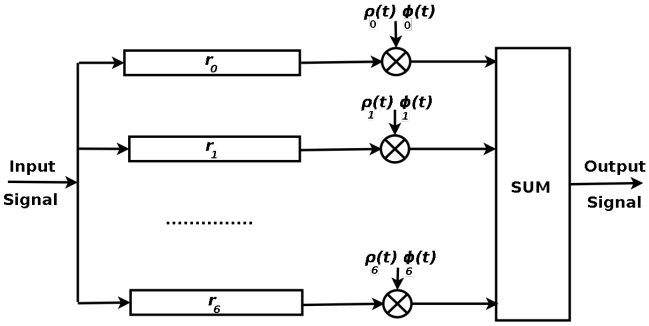

Figure 7: Multi-tap fading generator.

3.5 Software

WiNeTestEr is a distributed system with the Control PC, WARP boards and the UUTs being its components. The software slices running on each of these components are shown in Figure 3. We select the most significant slices and describe them in more details in the following paragraphs.

The different components of the system use a common *Communication Library* to exchange messages. We wrote it in both Java and C++. We used the Boost thread and system libraries [16] to maintain cross-architecture and cross-platform compatibility in C++. The library uses User Datagram Protocol (UDP) to provide services to send and receive messages in both blocking and non-blocking modes. Reliability is provided using optional acknowledgments and retransmissions on top of UDP. Communicating entities are identified by a unique ID, port and sub-system ID. Each module using the communication library creates an instance of it with its identity and a separate thread handles all communication responsibilities.

Logging is one of the integral part of our system. It is used for both debugging and collecting results and statistics of the experiment. We have implemented a log library which is used by all other software modules to log system events. Log messages are output to standard output/error streams and a log file. The log library allows software modules to log events and stamp them with the identity of the software module, time and severity of events.

Experiment Control is the main coordinator of all the software modules. It is present in the Control PC and coordinates all activities from the time an experiment is created until it is completed. When the user creates an experiment topology using the GUI, experiment control interacts with Topology Mapper and determines the the availability of the desired number of sites and the interconnections between them necessary to correctly emulate the network topology specified by the user. Once the experiment is flagged feasible, the Topology Mapper transfers the experiment parameters (channel conditions) from the GUI to the Channel Emulation module running on the PowerPC core which later forwards it to the FPGA. If the user has a UUT image, Experiment Control transfers the image to the UUT by interacting with the UUT Management module. The UUT management module controls the start and end of the experiment. It is also responsible for collecting experiment logs from the UUT and transferring them to the Control PC. Experiment Control makes these logs available to the user through the GUI.

A *GUI* application has been developed for the end user to run experiments. This is a standalone application which authenticates the user and allows them to create, start and abort experiments. It is also used to load any UUT images and collect the experiment logs. The user can create the experiment topology using the GUI, selecting different UUTs and specifying all the communication constraints among the UUTs. This application communicates only with the Experiment Control module in the Control PC. A screenshot of the GUI is shown in Figure 8. As shown in the figure, the user can create different types of UUTs, communication links between them and specify experiment parameters. This UUT topology can be saved on disk which can be used when the experiment needs to be repeated. The GUI can also be used to view properties of the testbed like physical topology of the UUTs, available/allocated UUTs for experiments etc.

A *MySQL database* stores all information about the testbed, completed and ongoing experiments. It stores information about the physical topology and the different UUTs in the testbed. When the testbed is first assembled, information about the different RF links and UUTs are manually entered into the database. When users want to use their own UUTs, information has to be manually entered into the database. All RF links are directed and are represented by the tuple $<$*UUT A, WARP board Number, Port Number, UUT B*$>$ which captures the unidirectional connection from UUT A to UUT B through a RF board's output port. An RF link is considered available if and only if UUT A, UUT B and the WARP board on which the RF board is mounted are all up and running. Each of these components periodically send *Keep Alive* messages updating the database. Topology Mapper uses the requirements from an experiment and the information in the database to check the feasibility of the experiment. If it is feasible, it allocates RF links and UUTs to the experiment. Apart from this, the database has information about all authorized users of the system. When users first start the GUI, they are authenticated against the information in this database. *Data Access* is a library which wraps all the internals of database access and provides application specific primitives for the other software modules to access the database.

Topology of an experiment is restricted by the physical topology of the testbed and the presence of other concurrent experiments. The problem of mapping a set of links required by the user to the available physical connections in the testbed is an instance of Directed Subgraph Isomorphism. *Topology Mapper* uses an approximation algorithm to find a subgraph of the physical connections in the testbed that satisfies the user requirements. If such a subgraph is found, its links are exclusively allocated to the user experiment for the required duration. As explained above, the information about the available links and whether they are being used by other experiments or not, is retrieved from the database.

As the name suggests *UUT Management* controls the actions of the UUT. It accepts the execution image which consists of a user image and a start-up script having all the parameters, environment variables necessary for the execution of the user application. When the user uploads an execution image through the GUI, it is stored in the file server which is later fetched by the UUT Management using Trivial File Transfer Protocol (TFTP). It accepts commands to start, stop, abort an experiment from the user through the Experiment Control. It also informs the user about any premature termination of the user image. Once the experiment is terminated, it transfers the experiment logs back to the user. UUT Management communicates with the Experiment Control in the Control PC using the *Communication Library*.

Channel Emulation is responsible for transferring the experiment parameters from the user to the FPGA to get the desired channel conditions. Channel Emulation slice on the ControlPC reads the parameters from the user and writes it to the database. The Control PC sends the experiment ID to the channel emulation slice running on WARP boards allocated to this experiment by the Topology Mapper. Channel Emulation slice on the WARP board reads the emulation parameters from the database, converts them to the format required by the FPGA and writes them to a designated memory location.

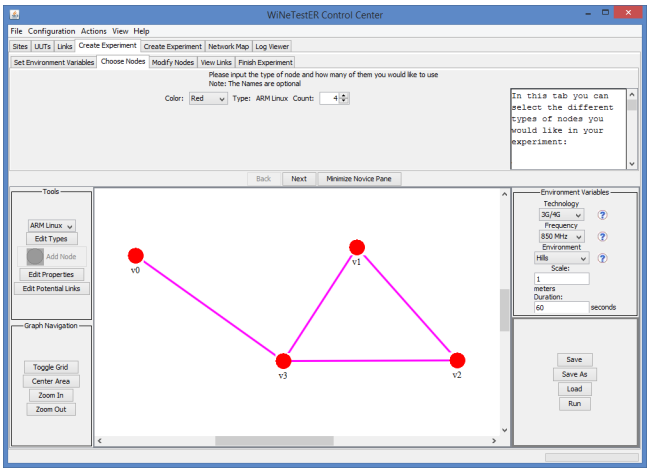

Figure 8: A screenshot of the GUI showing a topology of four UUTs.

4. EXPERIMENTAL VALIDATION

In this section we compare WiNeTestEr's channel emulation performance with a commercial channel emulator, Azimuth ACE MX MIMO [5]. It is one of the state-of-the-art channel emulators used by industry and academia to test complex wireless protocols. We chose not to perform any over-the-air experiments due to the difficulty in controlling the multi-path parameters. Even a seemingly simple environment like an open-air football field will have several taps (paths taken by different signal reflections).

We used two Ubiquiti SR-71 Cardbus WiFi adapters (Atheros AR9160 chipset) as UUTs. The cards were connected to two laptops running Linux 3.2 which includes the ath9k driver. Both cards were set to join an adhoc network operating on channel 14 (center frequency 2484 MHz) as per the IEEE 802.11 PHY/MAC standard for the 2.4 GHz ISM band. The basic rate of the adhoc network was fixed to 36 Mbps to avoid the results being affected by the driver's auto-rate algorithm. We could have picked another value for the fixed rate. However, we found 36 Mbps to provide a good balance between sensitivity to different channel conditions and the ability to gracefully degrade in performance as the channel worsens.

An experiment starts by the cards joining the adhoc network. Once the cards associate, we use Iperf [17] to send UDP packets carrying a 1470 byte payload from one machine to the other for 4 minutes while recording the average throughput achieved every second. At the end of the experiment, Iperf reports the number of packets lost during the session.

We defined four environments (channel conditions) each with a different number of taps. Table 1 shows the emulation parameters for Environment 1. It represent the basic case of having a single copy of the signal propagating through the channel. Environments 2, 3 and 4 use the top 2, 3 and 4 taps (respectively) of the ITU Vehicular - A Channel Model [18] shown in Table 2. The model uses a Doppler value of 184 Hz which results in the signal fading in a way roughly equivalent to that experienced by a vehicle traveling at 80 km/hr.

Doppler	1 Hz		
	Tap_Delay(ns)	Tap_Gain(db)	K-Factor(db)
Tap 1	0	0	-99

Table 1: Environment 1 channel parameters.

Doppler	184 Hz		
	Tap_Delay(ns)	Tap_Gain(db)	K-Factor(db)
Tap 1	0	0	-99
Tap 2	310	-1.0	-99
Tap 3	710	-9.0	-99
Tap 4	1090	-10.0	-99
Tap 5	1730	-15.0	-99
Tap 6	2510	-20.0	-99

Table 2: ITU Vehicular A Channel Model.

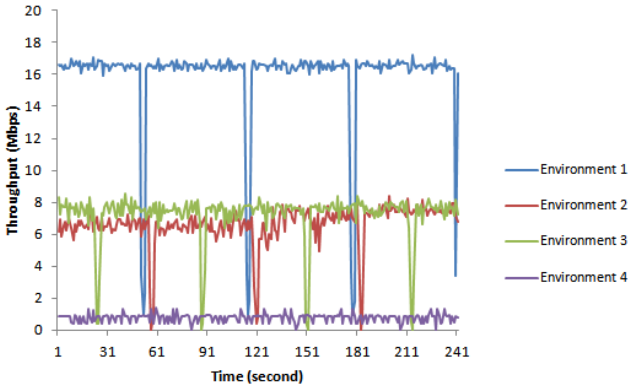

Figure 9: Throughput vs. Time plot for the experiments through WiNeTestEr.

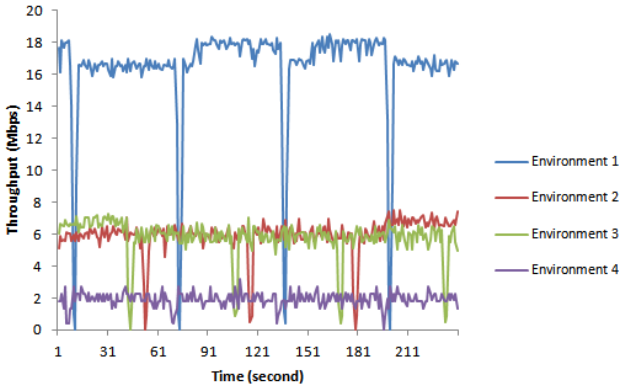

Figure 10: Throughput vs. Time plot for the experiments through Azimuth.

For each environment, we ran the experiment through both WiNeTestEr and Azimuth and recorded the throughput results. Figures 9 and 10 show the results obtained from WiNeTestEr and Azimuth, respectively.

4.1 Results Analysis

The results show a dip in the throughput roughly every minute. This happens with both WiNeTestEr and Azimuth. To isolate the issue, we ran the experiment over-the-air in an indoor interference-free environment. The cards experienced the same dips in throughput. While it would be interesting to dig deeper into this observation and find the root cause, it is out of the scope of this paper. We attribute these dips in throughput to the hardware or the driver of the adapters.

Both WiNeTestEr and Azimuth results share a trend of decreasing throughput with the increase of the number of taps. This matches the expectations we have for different environments especially environments 2,3 and 4 which emulate high mobility.

The results however are not identical. Despite sharing the same trend, WiNeTestEr and Azimuth achieve different throughput for the same channel parameters. We attribute this to the different level of control we have over the low level multi-path parameters in Azimuth compared to WiNeTestEr. Each tap in the multi-path parameters consists of a set of components. Unlike WiNeTestEr, Azimuth does not provide an interface to specify the value of each of these components. WiNeTestEr allows for repeating experiments under identical multi-path parameters even in terms of individual tap components. It is worth noting that specifying these components is optional and WiNeTestEr provides a set of default components for unfamiliar users.

5. CONCLUSION AND FUTURE WORK

Winetester is a scalable and cost-effective channel emulator that allows researchers to perform accurate, repeatable and complex experiments (topology, multipath effects). The distributed design of the system is evident in both the hardware and the software architecture. The hardware consists of UUTs, RF boards and combiner boards, interconnected to form the base topology. The software consists of a central Control PC that communicates with embedded Linux running on WARP boards and the FPGA that manipulates the signal using DSP. The paper discussed the different design and implementation aspects of each of the components.

Experimental results show that performance of WiNeTestEr is comparable to commercially available solutions. Being remotely accessible makes it an efficient alternative for researchers in academia and the industry alike.

WiNeTestEr is currently limited to operating in the 2400-2500 MHz band due to the RF board design. We intend to address this in the future revision of the RF board to be able to emulate environments in the range of 700 MHz to 6 GHz. Old RF boards will be swapped with the new ones without requiring any modification to the rest of the system. This will allow testing a wider range of devices and technologies (GSM, CDMA, LTE, 5 GHz Wifi, etc.).

6. ACKNOWLEDGMENT

We thank Krypton for their input in the design of RF and Combiner boards. We also thank Amba Kalur for her support during the initial phases of the RF board design.

7. REFERENCES

[1] Ryan Burchfield, Ehsan Nourbakhsh, Jeff Dix, Kunal Sahu, S Venkatesan, and Ravi Prakash. RF in the jungle: Effect of environment assumptions on wireless

experiment repeatability. In *IEEE International Conference on Communications*, pages 1–6. IEEE, 2009.

[2] NS-3. Network Simulator 3. `http://www.nsnam.org/`.

[3] David Cavin, Yoav Sasson, and André Schiper. On the accuracy of MANET simulators. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 38–43. ACM, 2002.

[4] Ehsan Nourbakhsh, Jeff Dix, Paul Johnson, Ryan Burchfield, S Venkatesan, Neeraj Mittal, and Ravi Prakash. ASSERT: A wireless networking testbed. In *Testbeds and Research Infrastructures. Development of Networks and Communities*, pages 209–218. Springer, 2011.

[5] Azimuth. ACE MX MIMO Channel Emulator for Broadband Wireless. `http://www.azimuthsystems.com/products/ace-channel-emulators/ace-mx/`, fetched April 2014.

[6] octoScope. octoBox MPE (Multi Path Emulator) Wireless Testbed. `http://www.octoscope.com/English/Products/octoBox_MPE/octoBox_MPE.html`, fetched April 2014.

[7] Kevin C. Borries, Glenn Judd, Daniel D. Stancil, and Peter Steenkiste. FPGA-based channel simulator for a wireless network emulator. In *69th Vehicular Technology Conference, VTC*, pages 1–5. IEEE, 2009.

[8] WARP Project. `http://warpproject.org`.

[9] Beagle Board. Beagle Bone Black. `http://beagleboard.org`.

[10] R.H. Clarke. A statistical theory of mobile-radio perception. *Bell Syst. Tech. J.*, 47:957 –1000, 1968.

[11] W.C. Jake. *Microwave Mobile Communication*. Wiley-IEEE Press, Piscataway, NJ, 1974.

[12] A. Alimohammad and B.F. Cockburn. Modeling and hardware implementation aspects of fading channel simulators. *IEEE Tran. on Vehicular Technology*, 57(4):2055 –2069, Jul 2008.

[13] Chengshan Xiao et al. Novel sum-of-sinusoids simulation models for rayleigh and rician fading channels. *Wireless Communications, IEEE Tran. on*, 5(12):3667 –3679, Dec 2006.

[14] Jianguo Xing et al. FPGA-accelerated real-time volume rendering for 3d medical image. In *3rd Int. Conf. on Biomedical Engineering and Informatics (BMEI)*, volume 1, pages 273 –276, Oct 2010.

[15] Pengda Huang, M.J. Tonnemacher, Yongjiu Du, D. Rajan, and J. Camp. Towards scalable network emulation: Channel accuracy versus implementation resources. In *Proceedings of IEEE INFOCOM*, pages 1959–1967, April 2013.

[16] Boost C++ Libraries. `http://www.boost.org`.

[17] Iperf. `http://iperf.fr/`.

[18] WP2.1 SUIT Project Deliverable. Fixed and Mobile Channel Models Identifications, July 2006.